**Declarations and Access Control pag 49**

identifiers son clases, variables y metodos

Inheritance, reutilizacion de codigo superclase(no sabe nada de las clases q heredan de este) y subclases(pueden sobreescribir metodos)

Interfaces, son superclases abstractas que definen metodos para subclases pero no impletan el soporte

cohesivo. Eso significa que cada clase debe tener un conjunto bien definido de responsabilidades

**identifiers** no pueden empezar con digitos, la primera letra y luego pude ser digitos letras o caracteres, usar CamelCase (convenciones de java)

nombre de clases inicia con Mayuscula, nombre de metodos y varieble inicia con minuscula

variables static and final todo mayuscula

**Keywords** no se pueden usar como identifiers

Importaciones

1. importamos paquetes
2. importamos librerias
3. existe solo una clase publica q debe llevar el mismo nombre q el archivo

**javac,** compilador de java , options and source files

**java,**invoca a JVM

**Main**

el metodo main() es usado por el JVM para ejecutar programas de java y puede ser **overloaded**

**Importaciones de clasee estaticas**

se importa con Import static, se debe cuidar de que no haya variables ambiguas, se pueden importar metodos staticos y variables constantes.

**Class Declarations and Modifiers**

modificadores de acceso(public , protected , private ) o default

Nonaccess modifiers (including strictfp , final , and abstract )

**Acceso a clases**

1. crear instancia de una clase dentro de otra
2. heredar de una superclase
3. aceder a metodos y variables

cuando se usa el **default** la clase solo es vista por otras clases del mismo paquete

**public** es un modificador q da acceso a q clases de otros paquetes puedan ver esta clase, importando el paquete, ademas se puede combinar modificadores public + final jamas debe ser abstrac + final

strictfp, se usa como modificador de clase o metodo, pero no para variables, cualquier metodo o cloase declarado con strictfp se alinea al estándar IEEE 754, el manejo de punto flotante y no depende de la plataforma

**final class** no permite herencia y se usa para asegurar q la implementacion no tenga cambios

abstrac class no puede ser instaciada, crea metodos q pueden ser sobreescritos por las clases q heredan (comportamiento especial)